ASP.Net Web Forms Course

الدرس الثالث

مفاهيم وأدوات

المحتويات :-

- **كيف يتم تنفيذ صفحة aspx .**
- **عملية إخراج ناتج للمتصفح (Rendering) .**
- **التفاعل مع بقية الآدوات .**
  - **Table**
  - **Literal**
  - **Calendar**
  - **FileUpload**
  - **Wizard**
  - **Panel**
  - **PlaceHolder**
  - **MultiView & View**
  - **AdRotator**
  - **Substitution**
  - **ImageMap**

الهدف :-

التعرف على كيفية تنفيذ الصفحة , ومن ثم التعرف على عملية إخراج الناتج إلى المستخدم تلك العملية التى تعرف بإسم Rendering , ثم ننتقل سوياً إلى إكمال ما بدأناه الدرس السابق من التعرف على الآدوات الموجودة بداخل ASP.Net .

كيف يتم تنفيذ صفحة aspx

فى بداية الآمر يقوم المستخدم بطلب الصفحة , وليكن على مثل هذا العنوان http://site.com/default.aspx , فيتوجه الطلب إلى الخادم IIS ومن ثم يقوم الخادم بتوجيهه إلى ASP.Net Runtime لتحديد الصفحة المطلوبة ليقوم بمعالجتها وإرسال الرد إلى المتصفح مرة أخرى .

ففى أول طلب للصفحة على الإطلاق يتم ترجمة الصفحة ورؤية إذا ما كان بها من أخطاء ,ثم يتحويل الصفحة إلى Class ثم يتم تضمينها داخل ملف DLL , ليتم التفاعل مع ملف DLL فيما بعد ,بدلاً من الذهاب إلى ترجمة الصفحة مرة أخرى , ولتذهب الطلبات القادمة إلى DLL مباشرةً .

فيتم فحص العنوان أولاً , ومعرفة إسم الصفحة المطلوبة , وهنا لدينا default.aspx فتكون الـClass الناتجة عن هذة الصفحة بإسم default_aspx , وعليه فعند قدوم طلب الصفحة يتم إنشاء Object من هذة الـ Class ليتم التفاعل معه لإخراج ناتج للمستخدم ليتم عرضه فى المتصفح .

يختلف الآمر على حسب نوع الطلب القادم من المستخدم , وكذلك تختلف ردة الفعل التى تأتى إلى المستخدم , فقد تكون هذة أول مرة يقوم المستخدم بفتح الصفحة , أو تكون المرة الثانية , أى ان الصفحة بالفعل مفتوحة أمامه وقام بالتفاعل معها وقد يكون إرساله طلباً للصفحة ناتج عن الضغط على أداة Button داخل الصفحة .ولكن ما الذى يجعل ASP.Net Runtime يشعر بالفارق بين الأمرين ؟

عند طلب المستخدم للصفحة أول مرة , فإن المتصفح يرسل طلباً للصفحة بإستخدام الأمر HttpGet ,وعندما يكون الطلب ناتج عن تفاعل المستخدم مع الصفحة ,فإنه يقوم بإرسال الطلب بإستخدام الأمر HttpPost .

وعلى هذا فإن ردة الفعل تختلف إستناداً على هذا , فنجد أن الطلب المرسل يسمى لدينا HttpRequest , وبه نجد كافة المعلومات عن الطلب الحالى . ونستطيع أن نعرف نوع الأمر القادم سواءً كان GET أو POST أو غيرهم .

وبعيداً عما تراه أعيننا فإنه يتم الحصول على قيمة نوع الطلب من HttpRequest ووضعها فى الخاصية IsPostBack الموجودة فى الصفحة , وعلى هذا فإننا نستطيع فعل التالى :-

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        // loading something for the First Page Loadb ;
```

```
    }
}
```

لاحظ هنا أننا قمنا بإستخدام علامة التعجب وتدل على Not فى visual basic .

إستناداً على هذا , فإن الصفحة تتفاعل مع الطلب وتقوم بتنفيذ مجموعة من الآحداث والدوال المختلفة , بمعنى أن دورة حياة الصفحة تتغير بناءاً على الطلب القادم . فنجد على سبيل المثال , أن الطلب حينما يكون HttpPost , فإن الصفحة تقوم بتحديث ما يعرف بـ ViewState ووظيفتها الإحتفاظ بحالة الصفحة .

وفى كل الآحوال يتم إعطاء الأمر للدالة Render بالتنفيذ , وهى المسئولة عن إخراج محتوى Html للمتصفح . وإليك حديثاً حول عملية Rendering .

عملية إخراج ناتج للمتصفح (Rendering)

كلمة Rendering نعنى بها أن الصفحة تقوم بإخراج محتوى HTML للمتصفح , حيث تكون هذة العملية ناتج عن تنفيذ الدالة Render الخاصة بالصفحة , والتى بدورها تقوم بإستدعاء الدالة Render الخاصة بكل الآدوات المتداخله داخل الصفحة على التوالى .

وإذا أردت أن تقوم بالتفاعل مع هذة العملية , فيمكنك أن تقوم بعمل Override للدالة Render كما ترى :-

```
protected override void Render(HtmlTextWriter writer)
{
    base.Render(writer);
}
```

وقد تستخدم هذة الدالة حينما تقوم بإنشاء Control خاص بك غير الموجود لديك تلقائياً.

ولعلنا نتساءل ماذا عن أدوات Server Controls والتى هى مكتوبة بإستخدام ASPX والتى تشبه XML , كيف يفهما المتصفح ؟ _____ فى الواقع كل أداة Server Control تمتلك دالة تسمى Render والتى بدورها تقوم بتحويل هذة الإداة إلى HTML , ولكن هذا هذا الآمر مهماً لنعرفه ؟ _____ فى الواقع نعم , تحتاج أن تعرف إلى عنصر فى HTML تصبح أداوات Server Control عليه بعد تحويلها , فعلى سبيل المثال , إذا أردت أن تقوم بالتفاعل مع هذة الآداوت من خلال الخادم فلا تقلق بشأن هذا الأمر , ولكن ماذا إن أردت أن تتفاعل مع هذة الآدات من خلال JavaScript ومكتبتها JQuery فإن هذة اللغة لا تتفاعل إلا مع عناصر HTML .

فإذا قمنا بسحب وإلقاء Button من شريط الآداوت فى الصفحة ليكون على هذا الشكل:-

```
<asp:Button ID="Button1" runat="server" Text="Click Me" OnClick="Clicked" />
```

ولاحظ وجود الحدث OnClick , وكذلك الخاصية Text والخاصية Runat وإنظر على ماذا سيكون شكل هذة الآداة بعد تحويلها إلى HTML ليتم عرضها فى المتصفح .

```
<input type="submit" name="Button1" value="Click Me" id="Button1" />
```
ولاحظ غياب الحدث OnClick وكذلك Runat وكذلك Text وظهورValue بدلاً منها , ثم وجود الخاصية type والتى تحدد ان هذا العنصر من نوع Submit أى يتسبب فى حدوث Postback .

لذلك حين نريد التفاعل مع هذا العنصر وليكن من خلال JavaScript ونريد معرفة النص بداخله , فإننا سنقوم بطلب قيمة الخاصية Value بدلاً من Text .

ينطبق نفس الآمر على أداة TextBox والتى فى ASPX هكذا :-

```
<asp:TextBox ID="TextBox1" runat="server" ontextchanged="TextBox1_TextChanged">
</asp:TextBox>
```
ولاحظ هنا أيضاً وجود الحدث OnTextChanged وأيضاً الخاصية runat وكذلك لعلك تعرف أنه هناك إمكانية لضبط الخاصية Text . ولكن لنرى ماذا سيكون شكله بعد إرساله إلى المتصفح وتحويله إلى Html .

```
<input name="TextBox1" type="text" value="Some" id="TextBox1" />
```
تحول TextBox إلى Input مع وجود الخاصيةType والتى تحدد أنه text على عكس العنصر السابق كان submit فى أداة Button, ولاحظ غياب الحدث OnTextChanged , وأيضاً غياب الخاصية runat.

ومن هذا , نجد أحداث الآداة التى تعمل على الخادم والتى تسمى Server events لا يتم إرسالها إلى الـ Client .كذلك تغيب الخاصية Runat فى HTML , ويتم إستبدال آدوات Server Control إلى عناصر مقابلة لها فى HTML .

فنجد أيضاً أداة مثل Calendar و GridView وغيرهم يتم تحويلهم إلى عنصر Table فى HTML .

كذلك فإن أداة Menu يتم تحويلها إلى UL وبداخلها بالطبع Li , تلك العنصر المعروف فى HTML .

وعلى هذا , فإن مهمتك الآن إدراج بعض الآداوت ورؤية إلى ماذا يتم تحويلهم بعد إرسالهم إلى المتصفح , فإن هذا سيعود عليك بالنفع عند العمل مع هذة الآدوات من خلال CSS أو JavaScript .

قد ترى المزيد من أكواد JavaScript قد تم حقن الصفحة بها , فهناك أدوات Server-Control تحتاج فى عملها إلى JavaScript .

التفاعل مع بقية الآدوات

Table

هى مجرد أداة تقوم بإدراج جدول عادى جداً , ولكن هو Server Control , اى تملك التحكم فيه من خلال الServer من حيث التفاعل مع هذا الجدول برمجيا من إضافة وحذف وتحرير .

```
<asp:Table ID="Table1" runat="server">
   <asp:TableHeaderRow>
          <asp:TableHeaderCell>...........</asp:TableHeaderCell>
          <asp:TableHeaderCell>..........</asp:TableHeaderCell>
   </asp:TableHeaderRow>
   <asp:TableRow>
            <asp:TableCell>............</asp:TableCell>
            <asp:TableCell>............</asp:TableCell>
   </asp:TableRow>
   <asp:TableFooterRow>
            <asp:TableCell> ............</asp:TableCell>
            <asp:TableCell> ............</asp:TableCell>
   </asp:TableFooterRow>
</asp:Table>
```

وإليك هذا المثال للتوضيح

Name	Job	Picture
Ali	Programmer	
Ahmed	Trainer	

كود التصميم

```
<asp:Table ID="Table1" runat="server" BorderStyle="Double" BorderColor="Red"
       GridLines="Both">
    <asp:TableHeaderRow runat="server">
       <asp:TableHeaderCell runat="server">Name </asp:TableHeaderCell>
       <asp:TableHeaderCell runat="server">Job</asp:TableHeaderCell>
       <asp:TableHeaderCell runat="server">Picture</asp:TableHeaderCell>
    </asp:TableHeaderRow>
    <asp:TableRow runat="server">
       <asp:TableCell runat="server">Ali</asp:TableCell>
       <asp:TableCell runat="server">Programmer</asp:TableCell>
       <asp:TableCell runat="server"><img src="Blue hills.jpg" width="50" />
    </asp:TableCell>
   </asp:TableRow>
   <asp:TableRow runat="server">
       <asp:TableCell runat="server">Ahmed </asp:TableCell>
       <asp:TableCell runat="server">Trainer </asp:TableCell>
       <asp:TableCell runat="server"><img src="Sunset.jpg" width="50"
/></asp:TableCell>
   </asp:TableRow>
```

هل تعرف أيضاً انه يمكن إضافة جدول أو صف او خلية برمجياً .

إضافة صف للجدول السابق عبر الكود

```
// إنشاء صف
TableRow Myrow = new TableRow();
// إنشاء خلية
TableCell Mycell1 = new TableCell();
// ملئ الخلية بالبيانات
Mycell1.Text = "Amr";
// إضافة الخلية إلى الصف
Myrow.Cells.Add(Mycell1);
Table1.Rows.Add(Myrow);
```

طريقة إنشاء جدول برمجيا

```
Table MyTable = new Table();
```

إنشاء صف برمجياً

```
TableRow Myrow = new TableRow();
```

يليه إنشاء خلية برمجياً

```
TableCell Mycell = new TableCell();
```

ملئ الخلية بالبيانات برمجياً

```
Mycell.Text = "Some Data";
```

يليه إضافة الخلية إلى الصف

```
Myrow.Cells.Add(Mycell);
```

يليه إضافة الصف إلى الجدول برمجيا

```
MyTable.Rows.Add(Myrow);
```

يليه إضافة الجدول إلى الصفحة

```
form1.Controls.Add(MyTable);
```

إضافة صف فى مكان معين فى الجدول

```
Table1.Rows.AddAt(0, Myrow);
```

حذف جدول من الصفحة (بهذة الطريقة يمكن ان تحذف أى أداة)

```
form1.Controls.Remove(Table1);
```

حذف صف من جدول

```
Table1.Rows.Remove(Myrow);
```

حذف كل الصفوف

```
Table1.Rows.Clear();
```

حذف خليه من صف

```
Myrow.Cells.Remove(Mycell);
```

حذف صف برقمه وهنا نحذف الصف الأول

```
Table1.Rows.RemoveAt(0);
```

أهم الخصائص

الخصائص	وظيفتها
BackImageUrl	وضع صورة كخلفية للجدول
GridLines	لإظهار الخطوط الداخلية والخارجية للجدول
HorizontalAlign	لضبط المحاذاة الأفقية للخلايا
Rows	لاضافة صفوف
Cells	وهى لإضافة خلايا
Caption	إضافة عنوان للجدول

Literal

تشبه Label كثيراً ولكن تختلف عنه فى أنها يمكن ان تخرج لك كود يعرض للمستخدم

```
<asp:Literal ID="Literal1" runat="server"> الحمد لله </asp:Literal>
```

ما سبق يخرج نص عادى ,ويمكن أن تكتب هكذا أيضاً

```
<asp:Literal ID="Literal1" runat="server" Text="الحمد لله"></asp:Literal>
```

كما يمكن إخراج كود بإستخدام الخاصية Mode بهذه الطريقة (لاحظ هاتين العلامتين ' ')

```
<asp:Literal ID="Literal1" runat="server" Mode="Encode"
             Text='<img src="Blue hills.jpg" />' />
```

Calendar

أداة لإدراج التقويم , وتتغير تلقائياً ليعرض الشهور بلغة المستخدم والمحددة حيث أنه قابل للتفاعل مع عدة لغات تلقائياً , فإذا كانت لغة المستخدم هى العربية , تم عرض التقويم بالعربية , وأيضاً تختلف العربية بمصر عن غيرها من بلدان العربية . كذلك حينما تكون اللغة المستخدم هى الإنجليزية فيظهر التقوم باللغة الإنجليزية.

```
<asp:Calendar ID="Calendar1" runat="server" ></asp:Calendar>
```

الخصائص

وتقدم لك أداة Calendar مجموعة من الخصائص لتنسيق العنوان ,الأيام ,اليوم المحدد,أيام العطلة ,اليوم الحالى .

الأحداث

SelectionChanged

تستطيع من خلال هذا الحدث أن تتعرف على اليوم الذى قام المستخدم بتحديده فى التقويم المعروض وتنفذ بعض المهام بناءاً على إختياره عند الحاجة لذلك .

DayRender

من خلال هذا الحدث يمكن أن تتحكم فى كيفية عرض الخلية داخل هذا التقويم , فيمكن إضافة نص إلى الخلية أو القيام ببعض التنسيقات لها , ولكن أى خليه داخل هذا التقويم ؟ ____ هذا الحدث يتم تنفيذه مع كل الخلايا , أى عندما يهم التقويم بالظهور فإن هذا الحدث يتم تنفيذه مع كل الخلايا أى مع كامل الآيام ,

فمن خلاله يمكن أن تقوم بإلكشف إذا ما كان الخلية الحالية تحتوى على ايام عطلة أو ربما أول الشهر أو ربما هو اليوم الحالى وإليك مثال :-

```
protected void Calendar1_DayRender(object sender, DayRenderEventArgs e)
{
    if (e.Day.IsToday)
    {
        e.Cell.BackColor = System.Drawing.Color.Red;
    }
}
```

FileUpload

أداة تتيح للمستخدم تحميل ملفات على موقع ما , حيث توفر هذة الآداة إمكانية تخزين هذا الملف التى يتم تحميله فى مجلد على الخادم أو ربما فى قاعدة بيانات , ولاحظ أنه يتم تحميل الملفات بمساحة 4096 أى بما يعادل 4Mb وإذا زاد عن هذا الحد يتم إظهار عطل فى الصفحة . ولحل هذة المشكلة يتم تعديل هذا الإعداد فى ملف Web.Config وسيأتى هذا الأمر فى درس كامل عن هذا الملف لما يحيوية من أهمية كبرى للموقع .

الخصائص

الخاصية	وظيفتها
HasFile	تسمح هذة الخاصية برؤية إذا ما كان المستخدم قد حدد ملفاً للتحميل أم لا , حيث تعود بالقيمة True أو False .
FileBytes	يمكنك إستخدام هذة الخاصية فى تخزين محتوى الملف إلى قاعدة بيانات , حيث يمكنك تخزينها فى حقل Binary لإن هذا النوع يقبل Array من نوع Byte .
FileName	وبها تستطيع معرفة إسم الملف

PostedFile	من خلالها يمكنك التعرف على معلومات أكثر عن الملف الذى يتم تحميله مثال : حجمه ونوعه وكذلك محتواه بإستخدام الخصائص :-
	`FileUpload1.PostedFile.ContentType`
	`FileUpload1.PostedFile.ContentLength`
	`FileUpload1.PostedFile.InputStream`
	`FileUpload1.PostedFile.FileName`

الدوال

SaveAs

تقدم هذة الآداة دالة تمكنك من حفظ الملف الذى تم إختياره إلى مجلد على الخادم , حيث يتم تزويد هذة الدالة بمسار المجلد الذى نريد حفظ الملف بداخله .

Wizard

هى أداة تتيح للمستخدم ان يمر بعدد من الخطوات لإتمام أمر معين .

```
<asp:Wizard ID="Wizard1" runat="server">
   <WizardSteps>
       <asp:WizardStep ID="WizardStep1" runat="server" Title="Step 1">
       </asp:WizardStep>
       <asp:WizardStep ID="WizardStep2" runat="server" Title="Step 2">
       </asp:WizardStep>
    </WizardSteps>
</asp:Wizard>
```

ولديك إمكانية إضافة أكثر من خطوة (Step) هكذا :

```
<asp:WizardStep ID="WizardStep1" runat="server" Title="Step 1">
</asp:WizardStep>
```

وكذلك إمكانية تسمية أى خطوة منهم بالاسم الذى تختاره مثال (تأكيد الحجز) إضافة إلى بقية خصائص تلك الخطوة .

```
<asp:WizardStep ID="WizardStep1" runat="server" Title=" تأكيد الحجز ">
</asp:WizardStep>
```

خصائص الآداة Wizard

الخاصية	وظيفتها
ActiveStepIndex	للتحكم فى رقم الخطوة المعروضة
HeaderText	لكتابة نص عنوان للـ Wizard
DisplaySideBar	لعرض الشريط الجانبى

خصائص WizardStep

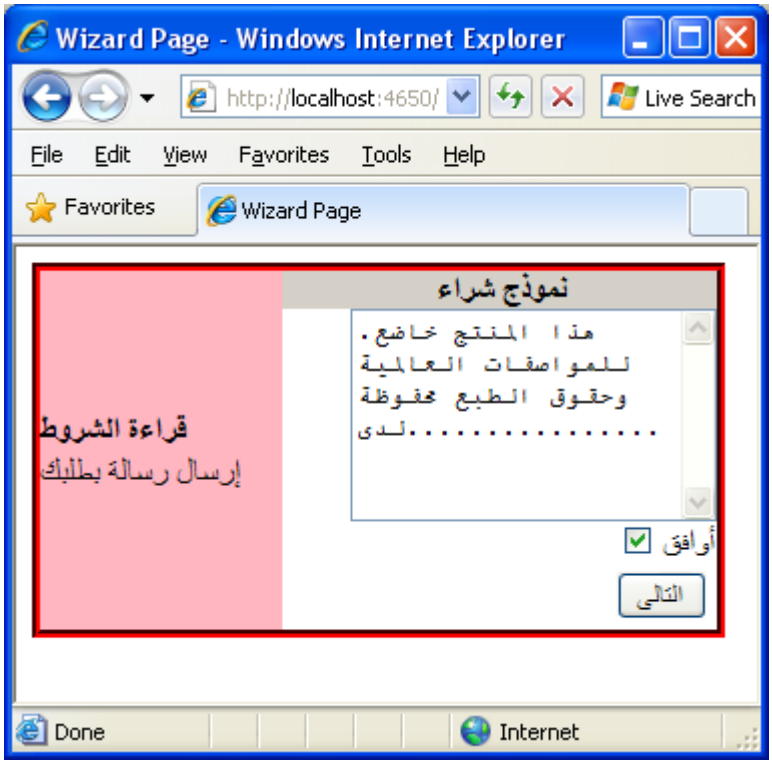
الخاصية	وظيفتها
AllowReturn	وتقبل القيمة True أو False وتحدد ما إذا كان فى الإمكان الرجوع للخطوة السابقة أم لا .
Title	ونحدد بها عنوان الخطوة
StepType	ونحدد منها نوع الخطوة , والمراد شكلها , أى ربما تكون خطوة نهائية أى تحتوى على Finish أو ربما Start , وهكذا.

الأحداث

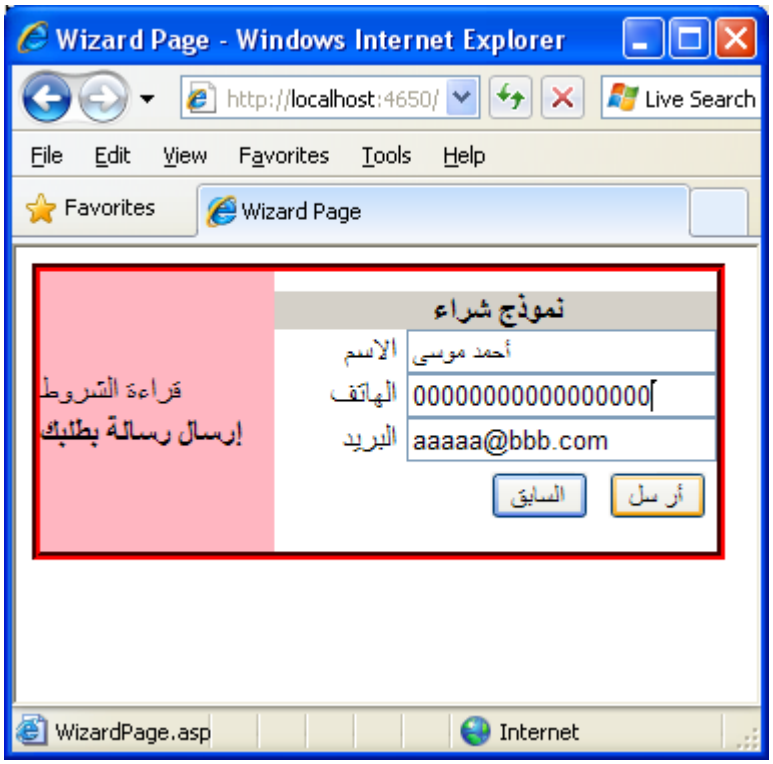
الحدث	وظيفته
ActiveInedxChanged	عند الإنتقال من خطوة إلى خطوة أخرى
CancelButtonClick	عند الضغط على Cancel
FinishButtonClick	عند الضغط على Finish
NextButtonClick	عند الضغط على Next
PreviousButtonClick	عند الضغط على Previous
SideBarButtonClick	عند الضغط على إسم الخطوة من الشريط الجانبى

مثال شكلى فقط بدون كود

الخطوة الأولى



الخطوة الثانية



XML Control

هى أداة لعرض ملف XML فى الصفحة , والمقصود هنا هو عرض النص الذى بداخل هذا الملف ,و إما أن يتم عرضه كنصاً عادياً بدون تنسيقات أو أن يتم عرضه فى شكلاً منسقاً ولكن حينها يمكنك إستخدام XSLT .

```
<asp:Xml ID="Xml1" runat="server" DocumentSource="~/XMLFile.xml" >
</asp:Xml>
```

الخصائص

الخاصية	وظيفتها
DocumentSource	نحدد هنا مسار ملف XML المراد عرض النص الذى به .
TransformSource	وهنا نحدد مسار ملف XSLT والذى من المفترض أن يحتوى على كيفية إخراج محتويات ملف XML بالتنسيقات المراده .
Document	يمكنك من خلالها تحديد XmlDocument من خلال الكود كما سترى.

ملئ الأداة من خلال كود C#

```
protected void Page_Load(object sender, EventArgs e)
    {
        XmlDocument Doc = new XmlDocument();
        Doc.Load(Server.MapPath ("XMLFile.xml"));
        Xml1.Document = Doc;
    }
```

Panel

هى أداة تسمى (Container) أى إطار . تحتوى على إما أدوات أو نصوص , والترجمة لها فى العربية "لوحة ".

```
<asp:Panel ID="Panel1" runat="server" Height="50px" Width="125px">
</asp:Panel>
```

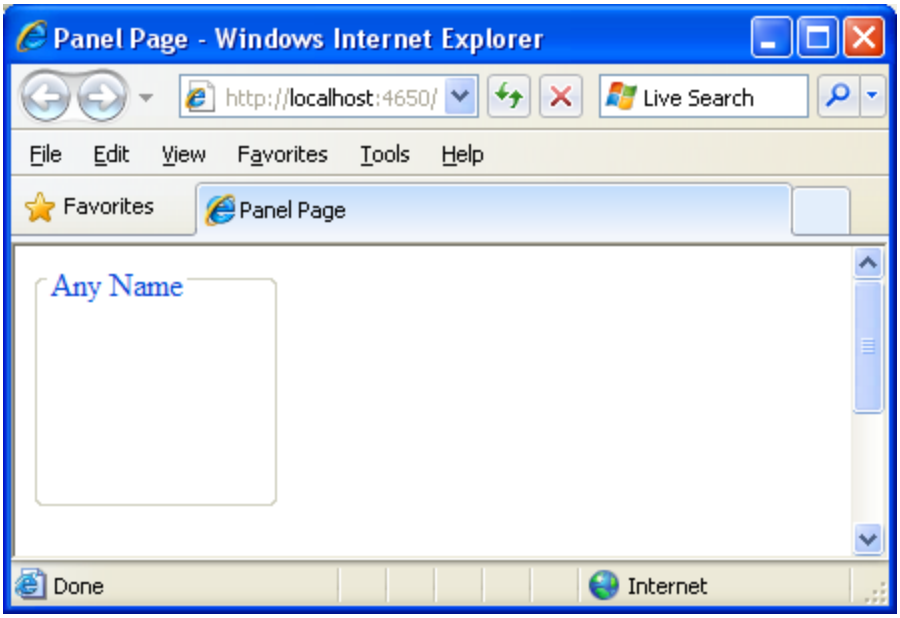
وعند إدارج اداة على سبيل المثال Button يكون هكذا :-

```
<asp:Panel ID="Panel1" runat="server" Height="50px" Width="125px">
   <asp:Button ID="Button1" runat="server" Text="Button" />
</asp:Panel>
```

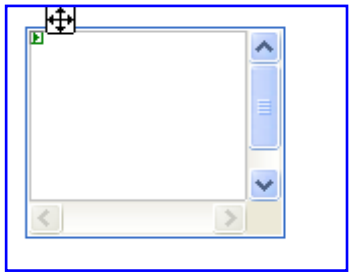
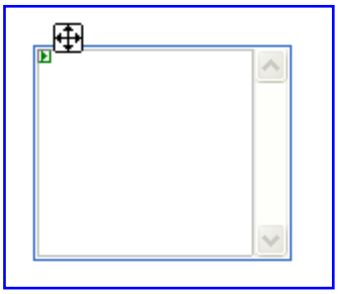
الخصائص

الخاصية	وظيفتها
Direction	ضبط الإتجاه من اليمين لليسار أو العكس
HorizontalAlign	ضبط المحاذاة الأفقية داخل الـ Panel (إتجاة النص - إتجاة الأدوات) و تكون (يسار – وسط – يمين – متساوى)
ScrollBars	لإظهار شريط التمرير داخل الـ Panel وستجد توضيح بالصور فى الأسفل
Warp	لإحتواء النص داخل الـ Panel
GroupText	لكتابة عنوان للـ Panel يظهر وكأنه نص لمجموعة

GroupText



Scroll Bar

Both الكل	Horizontal أفقى	Verticalرأسى
		

PlaceHolder

هى اداة محتوى أيضاً (Container) أى تضاف داخلها أداوت اخرى , وتشبه أداة Panel ولكن ليس لها جميع خصائصها , ويمكن أيضاً أن تقوم بإضافة أدوات بداخلها يدوياً أو برمجياً.

```
<asp:PlaceHolder ID="PlaceHolder1" runat="server"></asp:PlaceHolder>
```

كما يمكن أن نستخدمها فى إضافة الأدوات بواسطة الكود :-

```
protected void Page_Load(object sender, EventArgs e)
  {
      إضافةLabel //
     Label Label1 = new Label();
     Label1.Text = "الشمس غروب <br/>";
     PlaceHolder1.Controls.Add(Label1)
     إضافةImage //
     Image Image1 = new Image();
     Image1.ImageUrl = "~/Sunset.jpg";
     Image1.Width = 150;
     PlaceHolder1.Controls.Add(Image1);
}
```

MultiVie & View

هما أداتان يعملان بجانب بعضهما البعض , ولا يمكن لأحدهما أن تعمل بدون الآخرى , فهما مكملتان لبعضهما , والأصل فى الأمر , أن آداة MultiView ستحتوى بدورها على أكثر من آداة View كما ترى :-

```
<asp:MultiView ID="MultiView1" runat="server">
    <asp:View ID="View1" runat="server">
    </asp:View>
    <asp:View ID="View2" runat="server">
    </asp:View>
</asp:MultiView>
```

ولكن عند العرض للمستخدم لابد من عرض View واحد , ولكن على ماذا سيحتوى الـ View , أى ماذا سيكون بداخله ؟_____ هذا الأمر متروك لك , فيمكنك أن تضع بداخلها ما تريد من أداوات ونصوص .

الخصائص

الخاصية	وظيفتها
ActiveViewIndex	لتحديد رقم الـ view الذى تريد عرضه فى الصفحة بشكل مبدئي .

الأحداث

الحدث	وظيفته
ActiveViewChanged	وذلك عند الإنتقال من View إلى View من خلال الخاصية ActiveViewIdex وإعطائها رقم الـ View الذى نريد عرضه .

AdRotator

هى أداة نستخدمها لعرض إعلانات فى الصفحة ,حيث تتصل هذة الإداة بملف XML أو قاعدة بيانات, ففى حالة الإتصال بملف XML , فلابد وأن يكون لهذا الملف شكلاً خاصاً , وإليك الشكل الذى لابد وأن يكون عليه ملف XML :-

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
  <Ad>
    <ImageUrl>Image1.gif</ImageUrl>
    <AlternateText>Some Text </AlternateText>
    <NavigateUrl>mailto:ahmed_moosa83@yahoo.com</NavigateUrl>
  </Ad>
  <Ad>
    <ImageUrl>ali.gif</ImageUrl>
    <AlternateText>Web Developer</AlternateText>
    <NavigateUrl>mailto:ali@yahoo.com</NavigateUrl>
  </Ad>
  <Ad>
    <ImageUrl>Hossam.gif</ImageUrl>
    <AlternateText>Programmer </AlternateText>
    <NavigateUrl>mailto:hossam@yahoo.com</NavigateUrl>
  </Ad>
</Advertisements>
```

بعد ذلك تستطيع أن تجرى عملية الربط بين الآداة وبين هذا الملف السابق هكذا :-

```
<asp:AdRotator ID="AdRotator1" runat="server"
                              AdvertisementFile="~/XMLFile1.xml"  />
```

وبالتالى , ستتعرف الآداة على الصور والنصوص المرتبطة بها وكذلك العناوين المصاحبة للصور .

أما فى حالة وجود قاعدة بيانات , فإن هناك مجموعة من الخصائص لابد أن تقوم بضبطها , تراها فى الجدول التالى :-

الخاصية	وظيفتها
DataSource	للربط بين الآداة ومصدر البيانات (أى تحديد الجدول المراد داخل قاعدة البيانات , ويتم هذا الأمر بأكثر من طريقة)
DataSourceID	لتحديد إسم أداة DataSource التى تستخدمها وذلك فى حين أستخدام أحد أدوات DataSource مثال , SQLDataSource وAccessDataSource وغيرهما .
ImageUrlField	لتحديد إسم العمود الذى يحتوى على مسار الصور
AlternateTextField	لتحديد إسم العمود الذى يحتوى على النص الذى سيظهر عند تعثر إظهار الصورة فى المتصفح .
NavigateUrlField	لتحديد إسم العمود الذى يحتوى على روابط مرتبطة بالصور , بحيث سيتم الإنتقال إلى هذا العنوان عند الضغط على الصورة

Substitution

هذة الآداة لها طبيعة خاصة , حيث تعمل فى صفحة أو User Control قد تم بالفعل تخزينة داخل الـ Cache , فكما تعلمون ,أن البيانات التى تم تخزينها فى الـ Cache لا يتم تحديثها إلا إذا تم إنتهاء المدة الزمنية المحددة للتخزين فى الـ Cache , ولهذا , إذا أردنا أن نقوم بتحديث جزء من الصفحة المخزنة فى الـ Cache دون البقية , فإننا نستخدم أداة Substitution , وتنحصر وظيفتها فى تحديث جزء من صفحة أو User Control بعيداً عن الـ Cache .

ولكن طريقة عملها يختلف قليلاً عن بقية الآدوات الاخرى , حيث يتم تحديد دالة لها مواصفات خاصة , داخل هذة الدالة نضع الكود المراد تنفيذه لتحديث الجزء المراد .

فإذا قام المستخدم بطلب تحميل الصفحة فإنه سيتم تحميل الصفحة من منطقة الـ Cache إضافة إلى الذهاب إلى الدالة المحددة فى أداة Substitution لتنفيذها أيضاً .

```
<asp:Substitution ID="Substitution1" runat="server" MethodName="UpdatingData" />
```

فكما ترى تم تحديد إسم الدالة بإستخدام خاصية MethodNAme وهى كالتالى :-

```
static string  UpdatingData(HttpContext context)
   {
       return DateTime.Now.ToLongTimeString();
   }
```

ولاحظ أننا قمنا بإستخدام كلمة static والتى تساوى Shared فى Visual Basic .

ImageMap

يمكن إستخدام هذة الآداة لعرض صورة ولكن هذة الصورة يتم تجزئتها إلى أجزاء , بحيث يمكن التفاعل مع المستخدم إذا قام بالضغط على جزء محدد .

وتشبه هذة الصورة فى عملها كالتى توجد فى Facebook , حيث يتم وضع ما يعرف بـ Tag , فنلفترض أنه لديك صورة بها أربعة أشخاص وتريد أن تقوم بتجزئة الصورة إلى أربعة أجزاء , بحيث يظهر لك إسم الشخص الذى فى الصورة بمجرد الوقوف عليه فى الصورة من خلال المؤشر فى الشاشة , كل هذا يمكن أن تقوم به ImageMap .

كذلك ربما تكون لديك خريطة وتريد أن يقوم المستخدم بالضغط على مكان إقامته وبناءً عليه سيقوم الخادم بتنفيذ مهمه طبقاً لمكان إقامته. أيضاً هذا يمكنك فعله مع ImageMap .

```
<asp:ImageMap ID="ImageMap2" runat="server">
   <asp:CircleHotSpot X="200" Y="300" Radius="45" AlternateText="some1" />
   <asp:PolygonHotSpot Coordinates="177,0,400,0,400,223,335,154,127,180"
                       AlternateText="Some2" />
   <asp:RectangleHotSpot Left="200" Bottom="400" AlternateText="Some 3" />
</asp:ImageMap>
```

يمكنك تقسيم هذة الصورة إلى ثلاثة أشكال:- مستطيل , مربع , دائرى , كما ترى بالكود السابق , ويجب مع كل شكل تحديد إحداثياته , سواءً كان X,Y وLeft,Right وغيرهم .

الخصائص

الخاصية	وظيفتها
ImageUrl	مسار الصورة التى نريد العمل عليها .
HotSpotMode	نحدد هنا ماذا سيحدث عن الضغط على الجزء المحدد فى الصورة.
AlternateText	النص الذى سيظهر عند وقوف المستخدم على الجزء المحدد .
PostBackValue	القيمة التى يتم إرسالها إلى الخادم عند الضغط على الجزء المحدد فى الصورة عندما يكون HotSpotMode يأخذ القيمة PostBack .
NavigateUrl	المسار الذى سينتقل إليه المستخدم عند الضغط على الجزء المحدد فى الصورة عندما يكون HotSpotMode يأخذ القيمة Navigate .

الأحداث

Click

بالطبع عندما يقوم المستخدم بالضغط على المكان المراد فى الصورة , سيتم هذا الحدث , وعندها يمكنك فحص القيمة القادمة فى الخاصية PoskBackValue .

إنظر إلى الصورة الموجوة فى الصفحة بالأسفل :-

هؤلاء نفترض أن أسمائهم كالتالى Bobo و Bobe و Boba , ونريد أن نقوم بإظهار الإسم عند وقوف المستخدم على أحدهم , وكذلك عند القيام بالضغط على أحدهم يقوم بالذهاب الى الخادم ويأخذ معه قيمة , وبناءاً على هذا القيمة يفعل شيئاً ما .

فإليك الكود المستخدم لذلك :-

```
<asp:ImageMap ID="ImageMap1" runat="server" HotSpotMode="PostBack"
                                            ImageUrl="~/Penguins.jpg"
                                            OnClick="ImageMap1_Click"
                                            Width="600" >
   <asp:RectangleHotSpot Left="200" Bottom="400"
                                    PostBackValue="First"
                                    AlternateText="Boba" />
   <asp:RectangleHotSpot Left="400" Bottom="400"
                                    PostBackValue="Second"
                                    AlternateText="Bobe" />
   <asp:RectangleHotSpot Left="600" Bottom="400"
                                    PostBackValue="Third"
                                    AlternateText="Bobo" />
</asp:ImageMap>
```

ومن ثم عند الضغط على الجزء المراد يظهر التالى :-

```
protected void ImageMap1_Click(object sender, ImageMapEventArgs e)
 {
     switch (e.PostBackValue)
     {
         case "First":
             //do some thing
             break;
         case "Second":
             //do some thing
             break;
         case "Third":
             //do some thing
             break;
         default:
             break;
     }
 }
```